itk-elastix: Medical image registration in Python

Konstantinos Ntatsis†*, Niels Dekker‡, Viktor van der Valk‡, Tom Birdsong§, Dženan Zukic§, Stefan Klein‡, Marius Staring‡, Matthew McCormick§

Abstract—Image registration plays a vital role in understanding changes that occur in 2D and 3D scientific imaging datasets. Registration involves finding a spatial transformation that aligns one image to another by optimizing relevant image similarity metrics. In this paper, we introduce itk-elastix, a user-friendly Python wrapping of the mature elastix registration toolbox. The open-source tool supports rigid, affine, and B-spline deformable registration, making it versatile for various imaging datasets. By utilizing the modular design of itk-elastix, users can efficiently configure and compare different registration methods, and embed these in image analysis workflows.

Index Terms—medical imaging, image analysis, registration, elastix, ITK, wrapping, Python

Introduction

Image Registration

Image registration is a fundamental process in the field of scientific imaging that enables the alignment and comparison of images, facilitating the understanding of changes that occur within datasets. It involves finding a spatial transformation that optimizes relevant image similarity metrics, ensuring accurate alignment between images. A frequent registration type is the parametric approach where the spatial transformation is explicitly modeled. Examples of such transformation models are the rigid transform which allows translations and rotations, the affine transform that additionally includes shearing and the B-Spline transform that permits only local deformations. The reader can refer to Modersitzki [1] for an overview of the nonparametric registration. In addition to the parametric model, the choice of similarity metric plays a crucial role in the registration result and is dependent on the relationship of the pixel intensities between the images. Simple metrics such as normalized correlation are suitable for images with a linear intensity relationship, while more complex metrics such as mutual information [2] are employed for non-linear relationships.

Medical imaging heavily relies on image registration techniques [3] [4] to gain valuable insights and quantitative measurements. By registering medical images acquired at different time points or using various imaging modalities such as MRI and CT, researchers can analyze and quantify changes in anatomical structures, track disease progression and assess treatment efficacy. For instance, image registration allows the alignment of medical volumes across subjects to evaluate the impact of specific treatments, or the registration of sequential brain images to monitor tumor growth and response to therapy.

elastix

elastix [5] [6] is a well-known and widely used open-source toolbox dedicated to image registration. It provides a comprehensive range of algorithms and utilities designed for aligning images using diverse transformation models, similarity measures, and optimization strategies. One of its key strengths is its modular design, enabling users to easily configure and combine different registration methods to suit application-specific needs. Parameter files govern the registration process by specifying transformation models, similarity measures, optimization strategies, and related parameters. By customizing these configurations, users can seamlessly adapt elastix to their specific requirements, ensuring optimal registration outcomes.

For example, when aligning an MRI brain scan with a CT scan using elastix, users can configure the transformation model, such as an affine or B-spline transformation model, to capture the geometric relationships between input images. They can also specify the similarity measure, like mutual information or normalized correlation, to evaluate the quality of alignment. Additionally, users have the flexibility to adjust optimization strategies, including parameters like the maximum number of iterations, to fine-tune the registration process and achieve optimal results. elastix supports both the more typical pairwise registration but also groupwise registration [7] [8], where no image is specified as fixed but an implicit mean image is used instead as reference.

The elastix codebase is implemented in C++ and serves as an extension to the Insight Toolkit (ITK) [9]. Through nearly two decades of development, elastix has achieved a mature state, characterized by stability, practical effectiveness, maintainability, and general backward compatibility. ITK Image data structures play a crucial role within elastix, representing multidimensional pixel data augmented with spatial information. Acting as a vital link between the digital pixel space and the physical space of the imaged object, ITK Images facilitate accurate registration. By computing transformations that map points from the physical space of one image to corresponding points in another, elastix achieves precise and meaningful alignment outcomes within the physical space. Complementing elastix, a utility software named transformix was developed to enable the application of registration results to additional images.
The original and still-supported method to utilize elastix and transformix are command line executables. For the end user, this approach has the advantage that it does not require any external dependencies to be installed, which eases deployment. However, one limitation of this executable-based approach is its reliance on file input/output (I/O) operations. To address this limitation and enable more efficient in-memory operations, a C++ API was developed for elastix and transformix. This API follows the paradigm established by ITK and its processing filters. By adopting this design approach, elastix and transformix gained the ability to perform operations directly in memory. This enhancement provides users with greater flexibility and efficiency in their image registration workflows.

To further accommodate the needs of the users in the continuously developing scientific computing ecosystem, wrappings of the C++ code to other languages was developed in the form SimpleElastix [10], which still exists as part of the SimpleITK [11] package. More recently, we have embarked on developing a Python-specific wrapper called itk-elastix. This wrapper extends the functionality of elastix and offers an ever-expanding collection of Jupyter [12] examples, along with integration with other scientific processing libraries and visualization software. While there are other scientific python image image registration packages, itk-elastix stands out as a comprehensive Pythonic package with many image similarity metrics, implementations for 2D, 3D, and 4D images, and the ability to register a variety of imaging modalities. The subsequent sections of this paper delve into these aspects in greater detail.

**itk-elastix: Python wrapping**

The backend C++ elastix code is wrapped in Python with the Simplified Wrapper and Interface Generator (SWIG [13]). The Python wrapping of elastix, itk-elastix, brings the power of elastix to the Python ecosystem, providing effortless integration with other scientific processing libraries and visualization software. The itk-elastix Python package builds on the itk Python package’s pythonic interface and seamless integration with packages in the scientific Python ecosystem such as NumPy [14]. This enables users to leverage the rich functionality of elastix within their Python workflows, benefiting from its advanced image registration capabilities alongside popular Python libraries such as NumPy [14], SciPy [15], and MONAI [16] [17].

The process of updating and distributing the itk-elastix Python package is as follows: Once a significant number of changes have been made to the C++ elastix repository, a pull request is initiated in the itk-elastix repository to update its version. This triggers the CMake and Git tag update processes. The Continuous Integration (CI) system, which performs builds of Python packages across various Python versions (ranging from 3.7 to 3.11 at the moment of writing) and major platforms such as Windows, Linux, and macOS. When a git version tag is provided, the wrapped itk-elastix is automatically uploaded to PyPI, accompanied by a comprehensive summary of updates between the versions. As a result, users can easily install the latest itk-elastix by executing pip install itk-elastix within their Python environment. It is important to note that rigorous testing is conducted on the elastix backend functionality, with hundreds of tests performed during each pull request or commit, utilizing the CI system of the C++ repository. The test framework of elastix consists of various categories of tests, including low-level unit tests of the elastix library interface, minimal image registration tests on very small synthetic images, and larger regression tests of image registrations on realistic medical data. The tests are implemented using the CMake test driver CTest, the Python unitest module, and GoogleTest.

The Python wrapping for any ITK filter including elastix and transformix, offers two APIs: one functional and one object-oriented. We will describe the two API options and demonstrate the itk-elastix functionality with examples in the two following sections.

**Functionality**

*Registration/Transformation example*

The following example demonstrates the registration of 2D MRI brain images using the itk.elastix_registration_method and subsequent transformation of the corresponding moving mask using the itk.transformix_filter. The objective is to compare the overlap measure between the fixed mask and the transformed moving mask. It is important to note that this is a synthetic example where the fixed image intentionally exhibits significant deformations through an artificial non-linear transformation, solely for illustrative purposes. The masks utilized in this example represent segmentations of the head, including the brain and the skull. The procedure begins by reading the fixed and moving images from disk, followed by configuring a default set of B-spline registration parameters to be used for the registration process.

```python
import itk
from scipy.spatial.distance import dice
def main():
    # Load the moving and the fixed image from disk
    fixed_image = itk.imread('./data/fixed.mha', itk.UC)
    moving_image = itk.imread('./data/moving.mha', itk.F)

    # Configure a (default) parameter map with all the
    # registration parameters
    par_obj = itk.ParameterObject.New()
    par_map = par_obj.GetDefaultParameterMap('bspline')
    par_map.AddParameterMap(par_map)

    # Run the registration
    # 1. The Object Oriented way
    elastix_obj = itk.ElastixRegistrationMethod.New(
        fixed_image, moving_image)
    elastix_obj.SetParameterObject(param_obj)
    result_image = elastix_obj.GetOutput()
    rtp = elastix_obj.GetTransformParameterObject()

    # 2. The functional way
    result_image, rtp = itk.elastix_registration_method()

    # Load the corresponding masks
    fixed_mask = itk.imread('./data/f_mask.mha', itk.UC)
    moving_mask = itk.imread('./data/m_mask.mha', itk.UC)
```

Following the registration process, we load the masks from disk and apply the transformation parameters obtained during registration to the moving mask. To preserve the binary nature of the masks and avoid introducing interpolation artifacts, we utilize the nearest neighbor interpolator. This choice ensures that the binary properties of the masks are maintained throughout the transformation process.

```python
def main():
    # Load the moving and the fixed image from disk
    fixed_image = itk.imread('./data/fixed.mha', itk.UC)
    moving_image = itk.imread('./data/moving.mha', itk.F)
```

Following the registration process, we load the masks from disk and apply the transformation parameters obtained during registration to the moving mask. To preserve the binary nature of the masks and avoid introducing interpolation artifacts, we utilize the nearest neighbor interpolator. This choice ensures that the binary properties of the masks are maintained throughout the transformation process.

```python
# Load the moving and the fixed image from disk
fixed_image = itk.imread('./data/fixed.mha', itk.UC)
moving_image = itk.imread('./data/moving.mha', itk.F)
```
moving_mask = itk.imread('./data/m_mask.mha', itk.UC)

# Transform the moving mask using the result from the # registration
rtp.SetParameter(0, 'ResampleInterpolator', 'FinalNearestNeighborInterpolator')
result_mask = itk.transformix_filter(moving_mask, rtp)

# Compute dice on masks
initial_dice = 1 - dice(fixed_mask[:].ravel(), moving_mask[:].ravel())
result_dice = 1 - dice(fixed_mask[:].ravel(), result_mask[:].ravel())

print(initial_dice, result_dice)

The last part of the code above calculates the Dice coefficient between the fixed mask and the transformed moving mask by converting the pixel arrays in the ITK Images into NumPy array views and then call scipy.distance.dice() on them. The initial Dice score was 97.88% which increased to 99.37% after registration. Figure 1 visualizes the fixed, moving and result image as well as an overlay of the fixed image and the transformed mask.

Jupyter Notebook collection

In addition to the core registration and transformation functionality demonstrated above, itk-elastix offers other additional features. To help new users who are starting out, and also keep existing users up-to-date with the new feature implementations, we offer an evolving collection of Jupyter Notebooks as usage examples. Each of the Notebooks covers usually a specific topic, can be run independently, and includes comments and detailed explanations. The Notebooks are also tested automatically by CI with each pull-request or commit, and hence it is ensured that they always reflect the current API and functionality of the codebase. Such Notebooks include, but are not limited to:

- specifying masks or point sets for the registration
- transforming point sets and meshes
- groupwise registration
- logging options
- saving output to disk options
- reading/writing transform in hdf5 format
- calculation of spatial jacobian
- calculation of deformation field
- calculation of the inverse transform
- visualization of the registration

Interoperability with other packages

**ITK Transforms**

In addition to the fact that elastix is based on ITK, there is an ongoing effort to increase the compatibility between the two libraries even further. One particular example is the Transform classes [18]. In the following example, we show that ITK Transforms can be used directly by transformix:

```python
# Create an ITK (translation) transform
transform = itk.TranslationTransform.New()
transform.SetOffset([50, -60])

# Specify the image space of the transform
sp = moving_image.shape
parameter_map = {
    "Direction": ["1", "0", "0", "1"],
    "Index": ["0", "0"],
    "Origin": ["0", "0"],
    "Size": [str(sp[1]), str(sp[0])],
    "Spacing": ["1", "1"]
}
par_obj = itk.ParameterObject.New()
par_obj.AddParameterMap(parameter_map)

# Pass an ITK transform directly to transformix
transformix_obj = itk.TransformixFilter.New(moving_image)
transformix_obj.SetTransformParameterObject(par_obj)
transformix_obj.SetTransform(transform)
transformix_obj.Update()

# Get transformed (translated) image
translated_image = transformix_obj.GetOutput()
```

**NumPy & SciPy**

Interoperability with NumPy and, consequently, with SciPy libraries, comes from functionality in ITK to convert ITK Images to NumPy arrays and vice versa. The relevant code is:

```python
# itk image -> numpy array (deep copy)
image_array = itk.array_from_image(image_itk)

# itk image -> numpy array (shallow copy / view)
image_array = image_itk[:]

# numpy array -> itk image
image_itk = itk.image_from_array(image_array)
```

**Project MONAI**

More and more people work on the application of deep learning to medical imaging research. To that end, we developed
It has been extensively used and cited for over a decade, Documentation & reproducibility
input images and a transformed image at the center.

Fig. 2: The user interface of the elastix-napari plugin. For a
larger version of the image: https://github.com/SuperElastix/elastix-
napari#elastix-napari.

Integration with other software
Continuous efforts have been made to make itk-elastix
accessible to users of various tools. One notable community-driven
initiative is SlicerElastix, which seamlessly integrates elastix
(as an executable) into 3D Slicer [19] medical image visual-
ization software. In addition to this, recent endeavors focused
on developing the elastix-napari plugin for the Napari
[20] visualization software, which is written in Python. Figure 2
illustrates Napari user interface and showcases an itk-elastix
widget on the right side along with an example visualization of two
input images and a transformed image at the center.

Acknowledgment
We gratefully acknowledge the financial support received from
the Chan Zuckerberg Initiative (CZI) through the Essential Open
Source Software for Science award for Open Source Image Reg-
istration: The elastix Toolbox, numbers 2020-218571 and 2021-
237680 and the National Institute of Mental Health (NIMH) of
the National Institutes of Health (NIH) under the BRAIN Initiative
award number 1R|FM1|H126732.

Useful resources
- itk-elastix repository: https://github.com/InsightSoftwareConsortium/ITKElastix
- jupyter notebook examples: https://github.com/InsightSoftwareConsortium/ITKElastix/tree/main/examples
- elastix-napari plugin: https://github.com/SuperElastix/elastix-napari
elastix community forum: https://github.com/SuperElastix/elastix/discussions

parameter file model zoo: https://elastix.lume.nl/modelzoo/

elastix documentation and manual: https://elastix.lume.nl/doxygen/index.html

References


